

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

EXPERT SYSTEMS
DESIGNING ONTOLOGY WITH SWRL RULES
IN PROTÉGÉ

Tetiana Buzykina

Zagreb, January 2019

Table of Contents

Introduction	3
Theoretical Part	4
Practical Part	6
Building the ontology in Protégé 5.5	6
Reasoning	11
SWRL rules	15
Conclusion.....	16
Literature	17

Introduction

The main reason for my choice of the project topic is that despite the fact that ontologies have potential of further integration in various fields of Information Technologies and Artificial Intelligence in particular, their development still remains the topic of discussions in very narrow circles of IT specialists, so this is my contribution to popularizing ontologies. Beside of that, I already have some experience working with them during course “Knowledge management and ontology engineering” at Faculty of Information Technology at Taras Shevchenko National University of Kyiv.

Since the main purpose of creating ontology is to explain the connections between all the entities in a certain domain or even general knowledge, their scale may vary from primitive hierarchy with only 2-3 classes up to hundreds of thousands of entities. The domain of the built ontology is the course system at the Faculty of Electrical Engineering and Computing (FER) in particular, however I would assume that this may also be applicable to other faculties at University of Zagreb and to course structures at universities in other European countries in general with some minor changes in class relations.

Theoretical Part

In order to fully comprehend the place and importance of ontologies in IT, one should get to know about the Semantic Web.

In short, the Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners^[1]. It is based on the Resource Description Framework (RDF), which is widely used for ontologies development.

Ontology as a component of Semantic WEB structure:

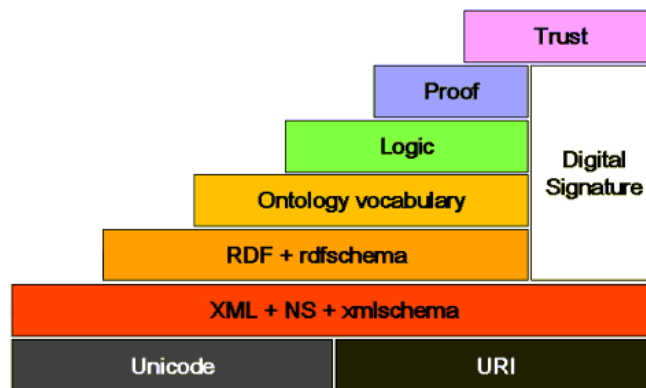


Image 1 – Semantic Web structure^[2]

What is ontology?

“...I use the term ontology to mean a *specification of a conceptualization*. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general. And it is certainly a different sense of the word than its use in philosophy”, - Tom Gruber, 1992 ^[3].

Conceptualization means an abstract simplified view of some selected part of the world, containing the objects, concepts, and other entities that are presumed of interest for some particular purpose and the relationships between them ^[4].

Simply designing ontology model includes the following stages:

- Decomposition (distinguishing the entities in the model)
- Identification (Creating individual entities)
- Classification (creating classes matching groups of entities, including the entities into the classes)
- Properties description (defining ways to express the information about characteristics and relations between entities)
- Values, connections (assigning values to entities, creating connections)^[5].

However, besides ontology model itself one should also take into consideration its implementation in chosen environment and also reasoning in order to be certain that there is no contradiction in ontology.

Practical Part

Building the ontology in Protégé 5.5

Designing the ontology model suggests that one has sufficient knowledge about the domain, hence the designer should do the research beforehand. My main source was UNIZG students network “Intranet”, where all the important information about the courses and the teaching staff can be easily found.

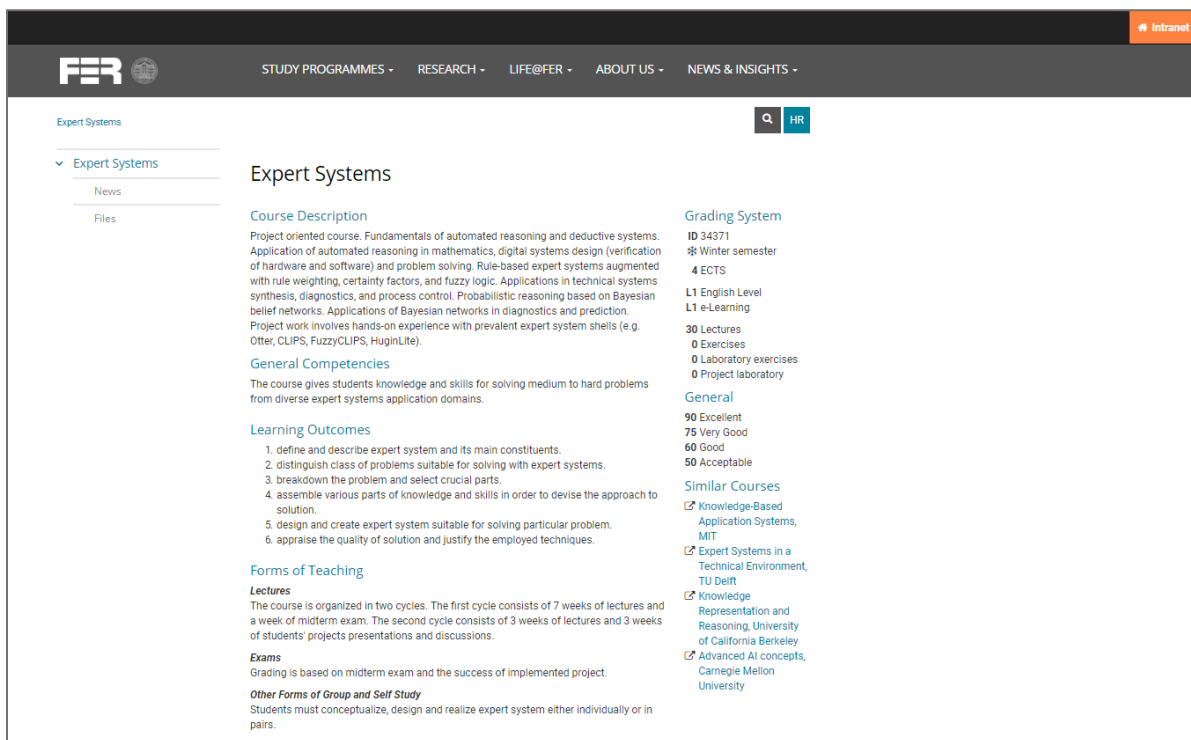


Image 2 – students network “Intranet”

The next stage is to build the one in the preferred environment. I have chosen Protégé (software developed by Stanford University) because it has relatively easy and user-friendly interface and offers variety of tools implemented, such as more than 3 different reasoners, SWRLQuery and SWRL rules editor, SPARQL editor etc.

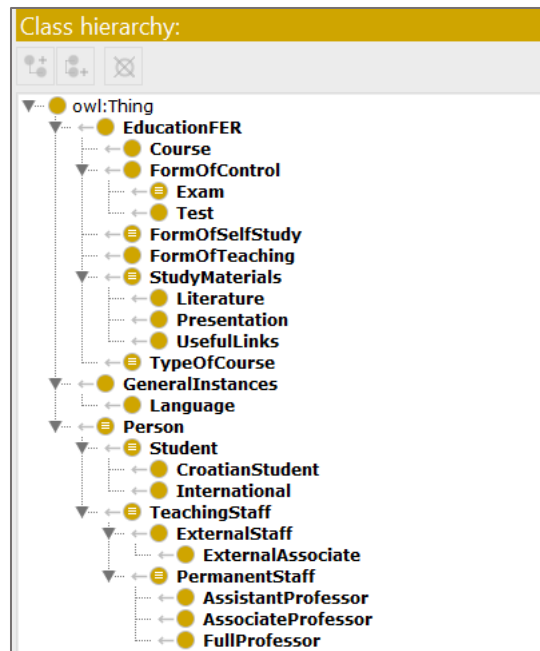


Image 3 – classes hierarchy

As it is shown on the image above, according to my ontology there are three main *disjoint classes*: “Person”, “Education FER” and “General Instances”, this means that a certain instance cannot belong to more than 1 of these classes at the same time. In terms of the ontology people involved in educational process can be either students or teaching staff. Since the ontology is created for Croatian university, I have assumed that students can be either from Croatia and speak only Croatian or both Croatian and English languages (that is represented by min 1 cardinality for object property “speaksLanguage”) or International (such as Erasmus students) and speak English language respectively. In order to avoid extra dependencies in the class hierarchy, I have put aside any language other than English and Croatian since I know that there are no courses taught in more than these two languages. Teaching staff can be either permanent or external. It is pretty obvious that a course should be necessarily taught by minimum 1 professor from permanent teaching staff.

Class “General Instances” contain entities that cannot be changed and neither depend nor interact with classes from “Person” or “Education FER”. In the given ontology it has a subclass called “Language” since language can only be used but does not interact with any other class actively.

Class “Education FER” is the biggest and the most important one since it describes the course structure – the main point of this ontology.

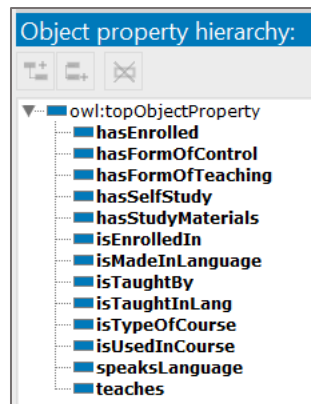


Image 4 – object properties for interaction between the classes

Now let us look at the usage of the properties above. Since we know that each course has a certain number of students enrolled, is taught by minimum 1 professor from permanent teaching staff and is of certain type such as “Bridge course”, “Skills Course” and etc. we can describe the mentioned dependencies using given object properties.

Description: Course

Equivalent To

SubClass Of

- (hasEnrolled min 1 Student)
 - and (hasEnrolled max 45 Student)
- (hasStudyMaterials min 1 Literature)
 - and (hasStudyMaterials min 1 Presentation)
- (isTaughtBy min 1 PermanentStaff)
 - and (isTaughtBy min 1 TeachingStaff)
- EducationFER
- hasFormOfControl min 1 FormOfControl
- hasFormOfTeaching min 1 FormOfTeaching
- hasSelfStudy some ({Project})
- isTaughtInLang min 1 Language
- isTypeOfCourse only ({'Bridge Course' , 'Elective Course' , 'Course for Successful Students' , 'Required Course' , 'Skills Course' , 'Specialization Course'})

General class axioms

- AssistantProfessor or AssociateProfessor or FullProfessor SubClassOf teaches min 1 Course
- CroatianStudent or International SubClassOf isEnrolledIn min 1 Course

SubClass Of (Anonymous Ancestor)

Instances

- CourseID34371
- CourseID72561

Target for Key

Disjoint With

- Student, FormOfTeaching, FormOfControl, Language, StudyMaterials, TeachingStaff, FormOfSelfStudy

Image 5 – “Course” relations with other classes from the ontology described with object properties

Description: Literature

SubClass Of +

- StudyMaterials

General class axioms +

SubClass Of (Anonymous Ancestor)

- (isMadeInLanguage value CroatianLang) or (isMadeInLanguage value EnglishLang)
- isMadeInLanguage min 1 Language
- isUsedInCourse min 1 Course

Instances +

- BookID000001
- BookID000002
- BookID000003
- BookID000004
- BookID000005
- BookID000006
- BookID000007
- BookID000008

Target for Key +

Disjoint With +

- UsefulLinks, Presentation

Image 6 – description of class “Literature”

Description: isEnrolledIn

Inverse Of +

- hasEnrolled

Domains (intersection) +

Ranges (intersection) +

- hasFormOfTeaching min 1 FormOfTeaching
- isTaughtInLang min 1 Language
- hasEnrolled min 1 Student
- hasStudyMaterials min 1 Presentation
- hasFormOfControl some FormOfControl
- (hasStudyMaterials min 1 Literature) and (hasStudyMaterials min 1 Presentation)
- isTaughtBy min 1 TeachingStaff
- EducationFER
- hasSelfStudy some FormOfSelfStudy
- isTypeOfCourse only ({'Bridge Course', 'Elective Course', 'Course for Successful Students', 'Required Course', 'Skills Course', 'Specialization Course'})
- Course
- hasStudyMaterials min 1 Literature

Image 7 – example of inverse object property “isEnrolledIn”

In fact, the intersection of all the ranges above will result the class “Course” and since in this ontology isEnrolledIn used only for “Student” domain class, the end triplet will be “Student” “isEnrolledIn” min 1 “Course”.

Reasoning

In order to check if ontology is built correctly (which means that at least classes, their instances, object and data properties do not contradict each other), we are able to use Reasoner. Protégé 5.5 offers an impressive variety of reasoners such as ELK, FaCT++, HermiT, Ontop, Pellet and more. The default reasoner is HermiT, but I have tried others as well.

I believe that the main advantage of using reasoner is that you are able to track down all the imperfections of the built ontology in tab “Class Hierarchy” – “Inferred”.

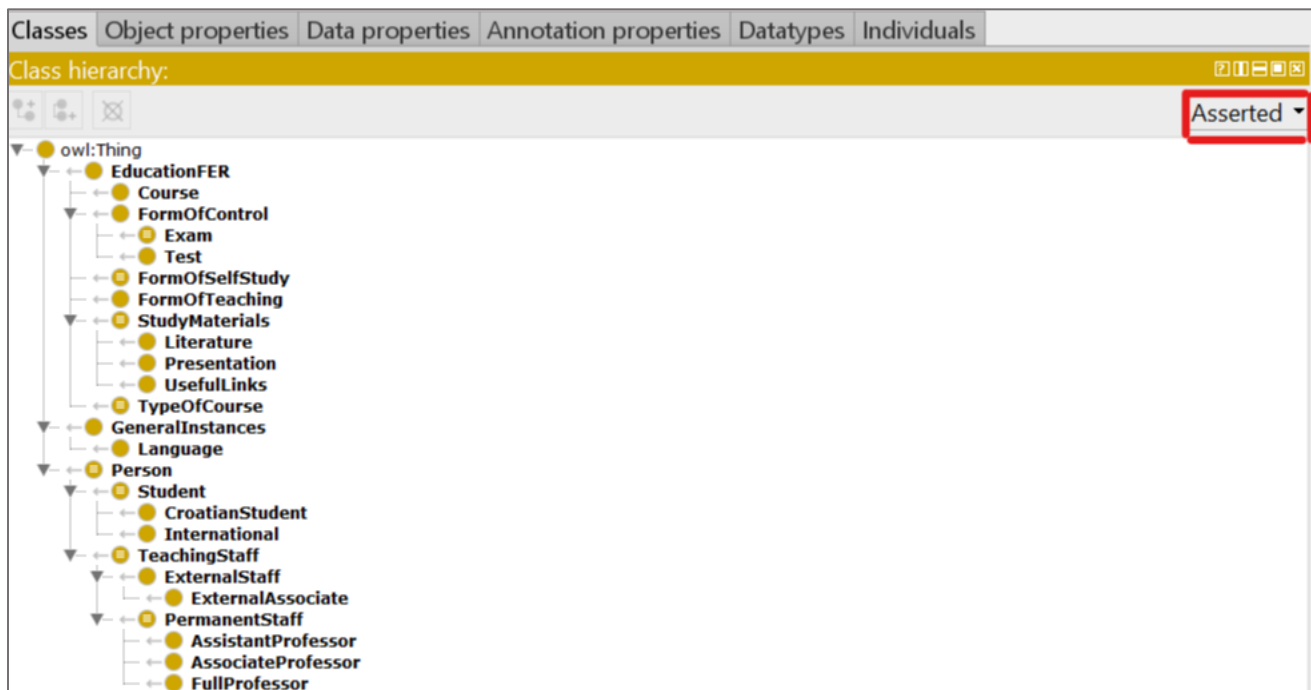


Image 8 – Asserted Class hierarchy

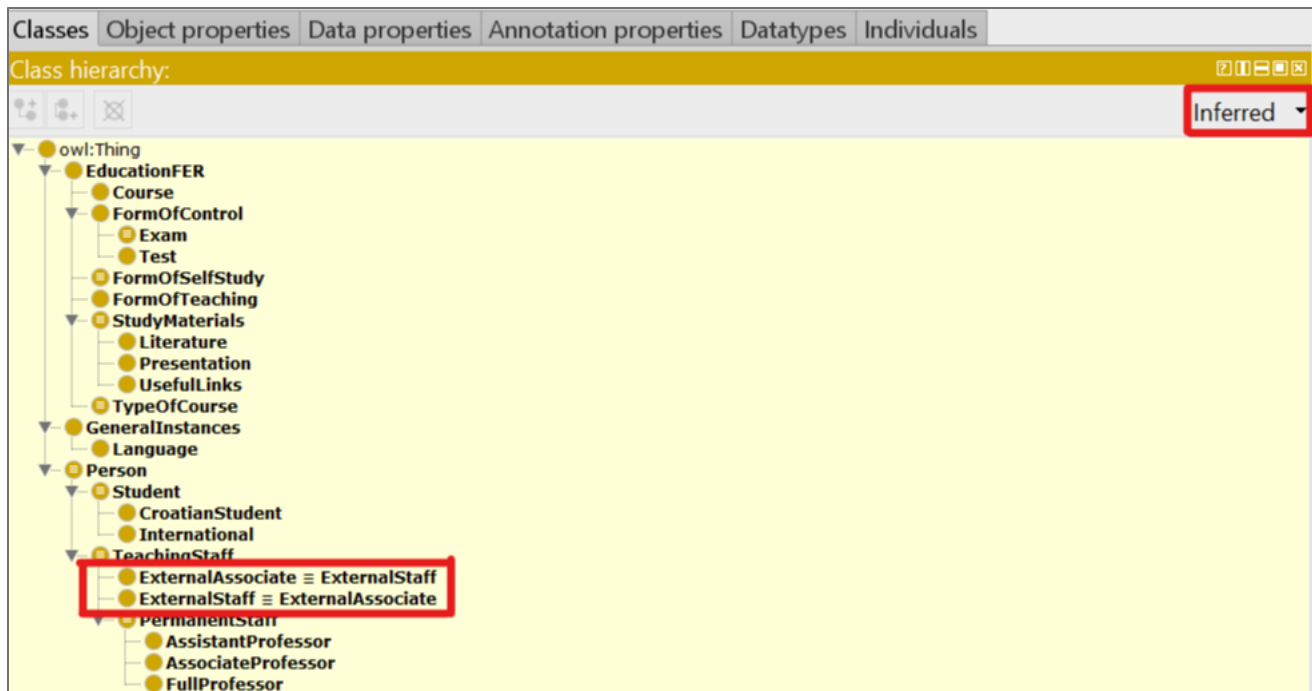


Image 9 – Inferred class hierarchy, reasoner logical conclusions

If an instance generates contradictions in ontology it is printed with red color.

Additionally, with reasoner turned on we are able to see “hidden” object properties that occur as the result of certain dependencies between classes or rules (for example SWRL rules). Thus, on Images 10 – 13 we are able to see that HermiT reasoner has assigned 2 implicit object properties “hasEnrolled” and “isTaughtBy” to the individual of class “Course” since they are inverse to “isEnrolledIn” and “teaches” object properties respectively. Additionally, despite the absence of any compiler for SWRL editor, the reasoner also checks if the ontology complies with SWRL rules and if so, also adds object properties and ontology axioms.

Class hierarchy: International

Annotations Usage

Annotations: StudentID0036520280

Annotations +

countryOfOrigin [type: xsd:string]
Ukraine

courseName [type: xsd:string]
Tetiana Buzykina

Description: StudentID0036520280

Types +

- CroatianStudent or International
- International
- isEnrolledIn min 1 Course
- Person
- speaksLanguage min 1 Language
- speaksLanguage value EnglishLang
- Student

Same Individual As +

Different Individuals +

Instances: StudentID0036520280

For: International

StudentID0036520280

Property assertions: StudentID0036520280

Object property assertions +

- speaksLanguage EnglishLang
- isEnrolledIn CourseID34371
- isEnrolledIn CourseID72561

Data property assertions +

Negative object property assertions +

Negative data property assertions +

Image 10 – object properties for the individual of class “International Student”

Class hierarchy: AssistantProfessor

Annotations Usage

Annotations: Alan_Jovic

Annotations +

countryOfOrigin [type: xsd:string]
Ukraine

courseName [type: xsd:string]
Tetiana Buzykina

Description: Alan_Jovic

Types +

- AssistantProfessor

Same Individual As +

Different Individuals +

Instances: Alan_Jovic

For: AssistantProfessor

Alan_Jovic

Property assertions: Alan_Jovic

Object property assertions +

- speaksLanguage CroatianLang
- teaches CourseID34371
- speaksLanguage EnglishLang

Data property assertions +

Negative object property assertions +

Negative data property assertions +

Image 11 – object properties for the individual of class “Assistant Professor”

Class hierarchy: Course

Annotations Usage

Annotations: CourseID34371

Annotations +

courseName [type: xsd:string]
Expert Systems

courseSchedule [type: xsd:string]
1. Course organization and administration. Introduction to expert systems. Brief outline of previous projects.
2. Symbol manipulation with LISP programming language.
3. Logic and fundamentals of automated reasoning. Structure of deductive systems.
4. Rule-based automated reasoning systems. Introduction to imperfect knowledge representation.
5. Representation and processing of imperfect knowledge. Introduction to probabilistic reasoning.
6. Probabilistic networks - 2.
7. Probabilistic networks - 3.
8. Midterm exam
9. Ontologies and modern knowledge based systems
10. Analysis of typical tools: Prover9/Mace4, FuzzyClins, Hugin lite, Protégé - 1

Description: CourseID34371

Types +

- (hasStudyMaterials min 1 Literature) and (hasStudyMaterials min 1 Presentation)
- Course
- EducationFER
- hasEnrolled min 1 Student
- hasFormOfControl some FormOfControl
- hasFormOfTeaching min 1 FormOfTeaching
- hasSelfStudy some FormOfSelfStudy
- hasStudyMaterials min 1 Literature
- hasStudyMaterials min 1 Presentation
- isTaughtBy min 1 TeachingStaff
- isTaughtInLang min 1 Language
- isTypeOfCourse only (('Bridge Course', 'Elective Course', 'Course for Successful Students', 'Required Course', 'Skills Course', 'Specialization Course'))

Property assertions: CourseID34371

Object property assertions +

- isTypeOfCourse 'Skills Course'
- hasFormOfControl Mid-termExam
- hasSelfStudy Project
- hasFormOfTeaching Lectures
- isTaughtInLang CroatianLang
- hasEnrolled StudentID0036520280
- hasStudyMaterials BookID0000003
- hasStudyMaterials BookID0000002
- hasStudyMaterials ExpertSystemsPres
- hasStudyMaterials BookID0000001
- hasFormOfControl FinalExam
- hasFormOfTeaching Consultations

Data property assertions +

Negative object property assertions +

Instances: CourseID34371

For: Course

CourseID34371

Same Individual As +

Image 12 – object properties for the individual of class “Course”, HermiT reasoner turned off

Class hierarchy: Course

Annotations Usage

Annotations: CourseID34371

Annotations +

courseName [type: xsd:string]
Expert Systems

courseSchedule [type: xsd:string]
1. Course organization and administration. Introduction to expert systems. Brief outline of previous projects.
2. Symbol manipulation with LISP programming language.
3. Logic and fundamentals of automated reasoning. Structure of deductive systems.
4. Rule-based automated reasoning systems. Introduction to imperfect knowledge representation.
5. Representation and processing of imperfect knowledge. Introduction to probabilistic reasoning.
6. Probabilistic networks - 2.
7. Probabilistic networks - 3.
8. Midterm exam
9. Ontologies and modern knowledge based systems
10. Analysis of typical tools: Prover9/Mace4, FuzzyClins, Hugin lite, Protégé - 1

Description: CourseID34371

Types +

- (hasStudyMaterials min 1 Literature) and (hasStudyMaterials min 1 Presentation)
- Course
- EducationFER
- hasEnrolled min 1 Student
- hasFormOfControl some FormOfControl
- hasFormOfTeaching min 1 FormOfTeaching
- hasSelfStudy some FormOfSelfStudy
- hasStudyMaterials min 1 Literature
- hasStudyMaterials min 1 Presentation
- isTaughtBy min 1 TeachingStaff
- isTaughtInLang min 1 Language
- isTypeOfCourse only (('Bridge Course', 'Elective Course', 'Course for Successful Students', 'Required Course', 'Skills Course', 'Specialization Course'))

Property assertions: CourseID34371

Object property assertions +

- isTypeOfCourse 'Skills Course'
- hasFormOfControl Mid-termExam
- hasSelfStudy Project
- hasFormOfTeaching Lectures
- isTaughtInLang CroatianLang
- hasEnrolled StudentID0036520280
- hasStudyMaterials BookID0000003
- hasStudyMaterials BookID0000002
- hasStudyMaterials ExpertSystemsPres
- hasStudyMaterials BookID0000001
- hasFormOfControl FinalExam
- hasFormOfTeaching Consultations
- hasEnrolled StudentID5739205672
- isTaughtBy Alan Jovic

Data property assertions +

Negative object property assertions +

Instances: CourseID34371

For: Course

CourseID34371

Same Individual As +

Image 13 – object properties for the individual of class “Course”, HermiT reasoner turned on

SWRL rules

The rules help set certain restrictions or make logical conclusions and computations in ontologies. However, one of the biggest disadvantages is that modern edition of SWRL does not have disjunction operator. Additionally, due to specificity of the chosen domain there is no such opportunity to perform calculations. Furthermore, the SWRL editor in Protégé 5.5 does not have user-friendly graphic interface from the previous versions of Protégé, which makes creating rules quite troublesome. On top of that, the majority of possible rules can be implemented via graphic entities editor which offers cardinality operators *min*, *max*, *exactly*, *only*, *some*, also conjunction operator *and*, disjunction operator *or*.

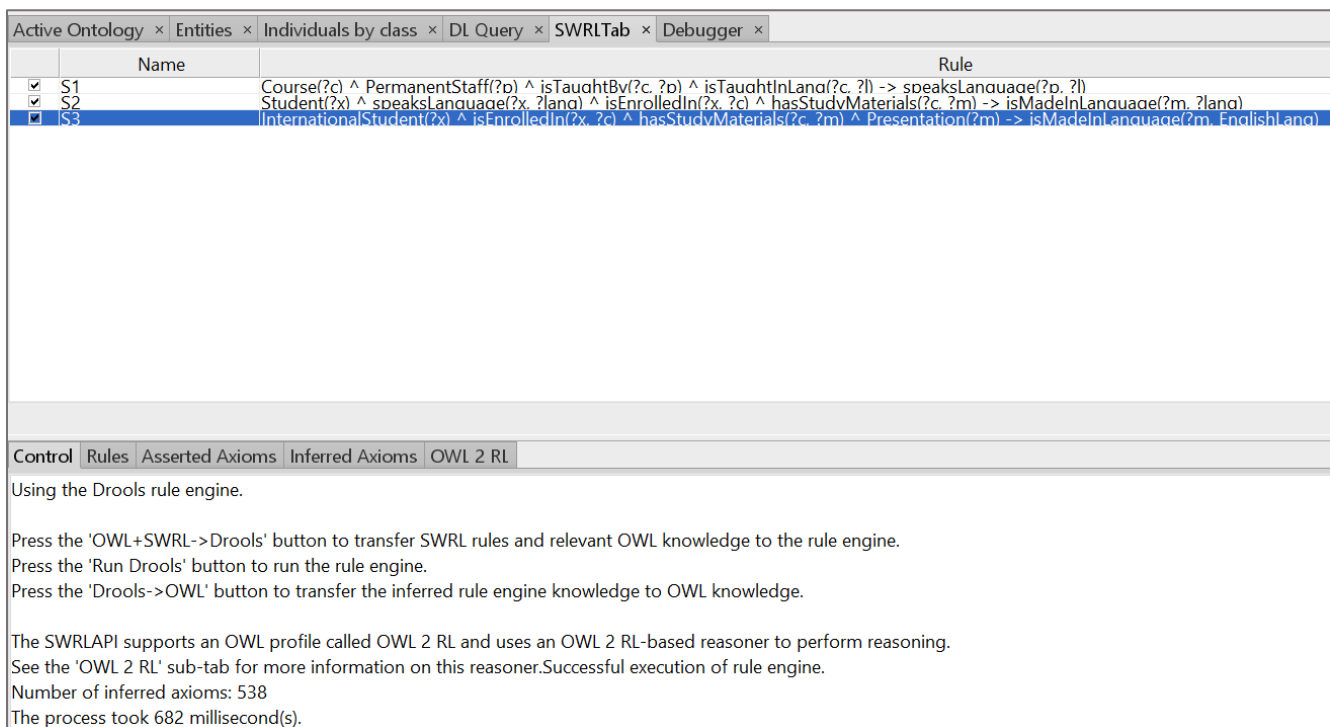


Image 14 – SWRL rules editor tab in Protégé 5.5

Conclusion

To sum up, this ontology could be published online with some minor changes. However, it took more than one week to come up with the class hierarchy for the given domain. Furthermore, I have encountered numerous difficulties while working with object and data properties. In order to make the ontology consistent I was exploring Stanford University's "pizza ontology" that can be downloaded straight to Protégé and is a great example of complex ontology.

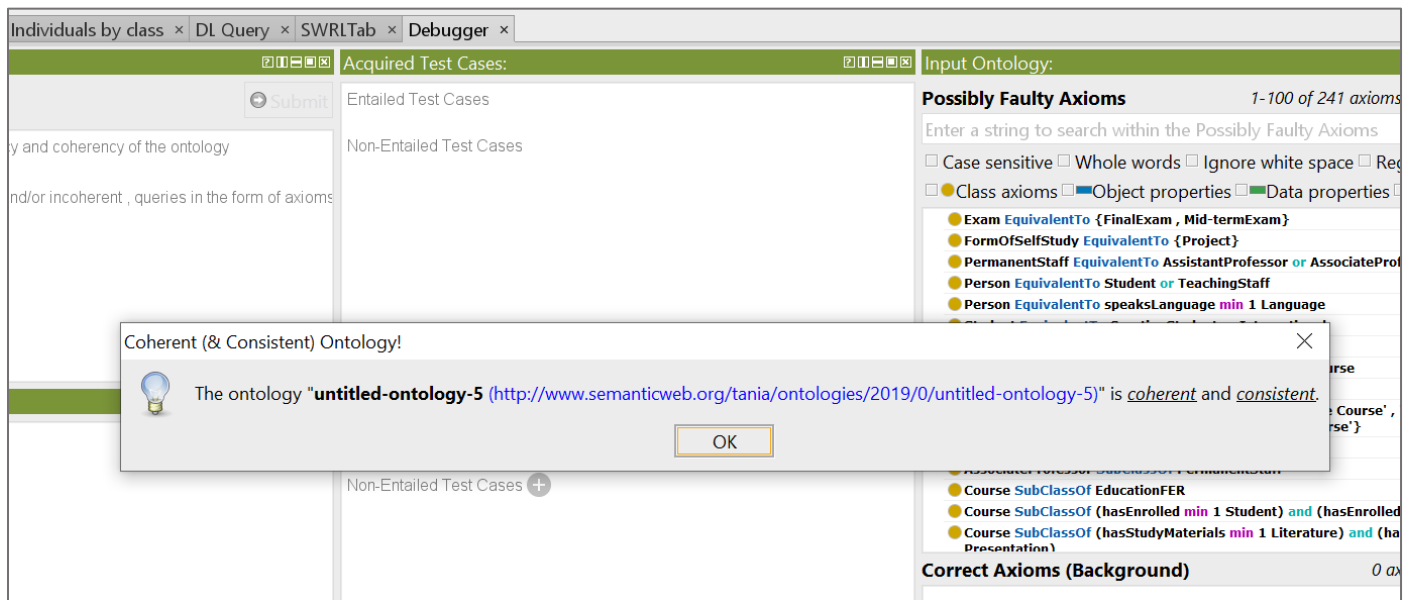


Image 15 – Debugging the ontology

Additionally, it may take time just to find proper documentation for designing ontology (aside from programmers forums such as StackOverflow or official Stanford University guide which is not adapted for the latest version of Protégé), which means that sadly it is still not popular enough among IT specialists.

Literature

1. <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>
2. [https://en.wikipedia.org/wiki/Conceptualization_\(information_science\)](https://en.wikipedia.org/wiki/Conceptualization_(information_science))
3. <https://www.w3.org/RDF/Metalog/docs/sw-easy>
4. <https://www.w3.org/2001/sw/>
5. Sergey Gorshkov – «Introduction to ontology modelling», 2016.